FortiAuthenticator - REST API Solution Guide

VERSION 4.0

FORTINET DOCUMENT LIBRARY

http://docs.fortinet.com

FORTINET VIDEO GUIDE

http://video.fortinet.com

FORTINET BLOG

https://blog.fortinet.com

CUSTOMER SERVICE & SUPPORT

https://support.fortinet.com

http://cookbook.fortinet.com/how-to-work-with-fortinet-support/

FORTIGATE COOKBOOK

http://cookbook.fortinet.com

FORTINET TRAINING SERVICES

http://www.fortinet.com/training

FORTIGUARD CENTER

http://www.fortiguard.com

END USER LICENSE AGREEMENT

http://www.fortinet.com/doc/legal/EULA.pdf

FEEDBACK

Email: techdocs@fortinet.com

2015-09-14

FortiAuthenticator 4.0 - REST API Solution Guide

23-330-264234-20150605

TABLE OF CONTENTS

Change Log

Date	Change Description
2014-20-20	Updated API to reflect changes since 3.2 release
2014-06-09	Updated API to reflect changes since 3.1 release
2013-11-18	Added missing group config in /localusers/
2013-10-21	Updated API to reflect changes since 3.0 release
2012-11-01	Updated API to reflect changes since 2.0 release
2012-07-12	Initial release

Introduction

This document introduces the FortiAuthenticator REST API and details how it can be configured and utilized.

Software versions

The API described within this document is as supported by FortiAuthenticator 4.0 and upwards.

Changes introduced in version 4.0

Several changes have been introduced in the API in version 4.0

- /auth: Implemented user lockout policy.
  - Specifying a password when authenticating a local user with only FTK authentication enabled (password-based auth is disabled) will now fail.
- /ldapusers:
  - LDAP users can now be edited or deleted
  - Add 2FA provisioning and integration features
- /localusers
  - Add expires_at and active fields to manage user's expiration time and account status, respectively.
  - Add 2FA provisioning and integration features
  - Add ftk_only field to control whether FortiToken-only authentication should be enabled.
  - Added support for the DELETE method
- /fgtgroupfilter:
  - sso_groups field is removed because SSO group is replaced by SSO filtering object
  - A new nested API to manage SSO filtering objects is added
  - new API added to /fgtgroupfilter/[id]/ssofilterobjects to manage FortiGate filter's SSO filtering objects
- /localgroup-memberships/
  - Add a new API to manage local group memberships. This API allows incremental update and deletion on group membership
  - Updated ambiguous PUT (list) method description and added an example PUT (list) request
  - Updated "General Response Codes" description

Changes introduced in version 3.2

Several changes have been introduced in version 3.2

- /auth:
  - Email/SMS token is now supported in /auth API
- /ldapusers:

    - Add /ldapusers API with a read-only access, to view user's token-based authentication configuration
    - Out-of-band token code (for email and SMS tokens) can be sent using the new /sendoobtoken resource
- /localusers:
    - API now returns an empty string (instead of null) for token_type field when user has no token-based authentication configured
    - Out-of-band token code (for email and SMS tokens) can be sent using the new /sendoobtoken resource
    - New filter expressions are added such as contains, icontains and in
- /fgtgroupfilter:
    - New filter expressions are added such as contains, icontains and in

The FortiAuthenticator API

Introduction to REST

An API (Application Programming Interface) is a set of defined interfaces to accomplish a task, such as retrieving or modifying data. FortiAuthenticator provides a Representational State Transfer (REST) API for interaction with components of the system. Programs communicate with the REST API over HTTP, the same protocol that your web browser uses to interact with web pages.

The REST API is based on interactions with a web page; data is treated like a static web page:

- Add data by POSTing a web page
- Fetch data by GETing a web page
- Update data by PUTing a web page
- Partial updates supported by PATCHing a web page
- Delete data by DELETEing a web page

After receiving the request, the FortiAuthenticator API sends back an HTTP response code. These error codes are summarized in Appendix A – API Response Codes.

Initializing the REST API

Unlike most other vendors, the FortiAuthenticator API is accessible without additional cost or licensing. The server however is disabled by default and needs to be configured.

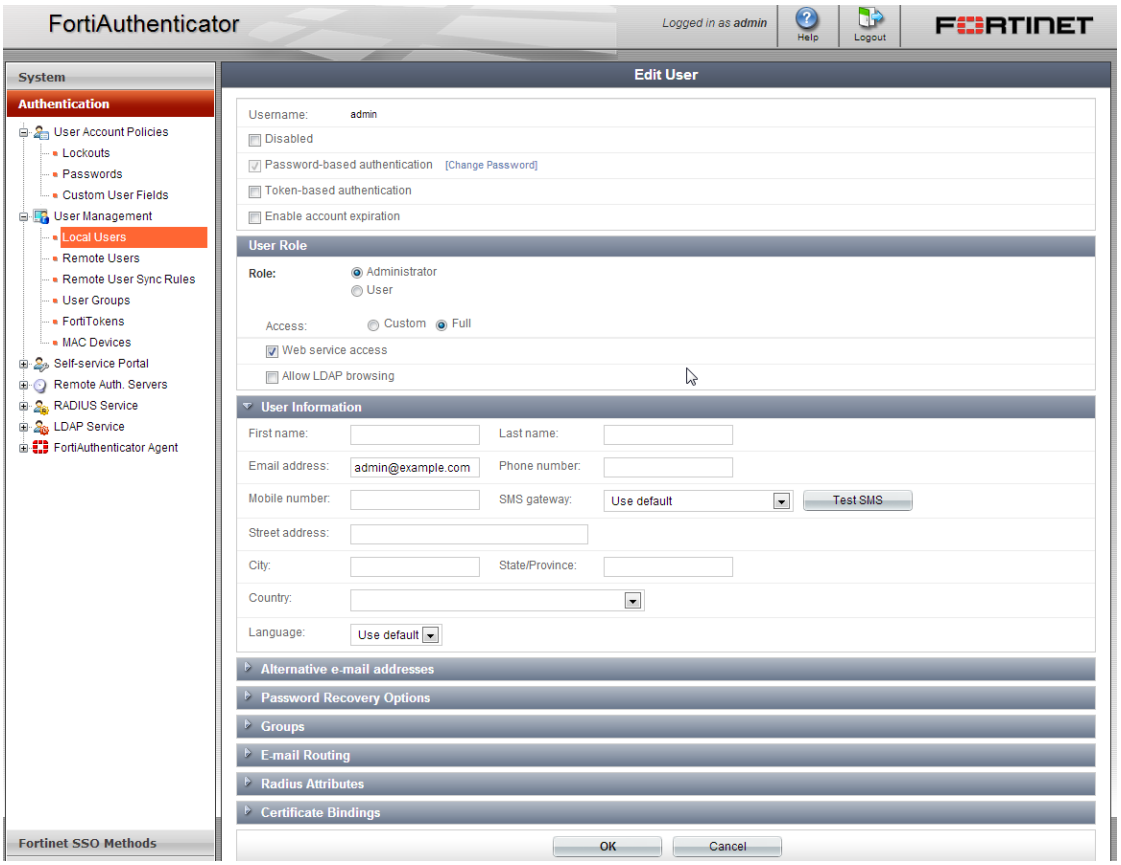
To enable the API, enable a user with administrator rights and select Web Service Access.

You must configure an e-mail address for the user at this point to as the API challenge key will be emailed to the address specified.

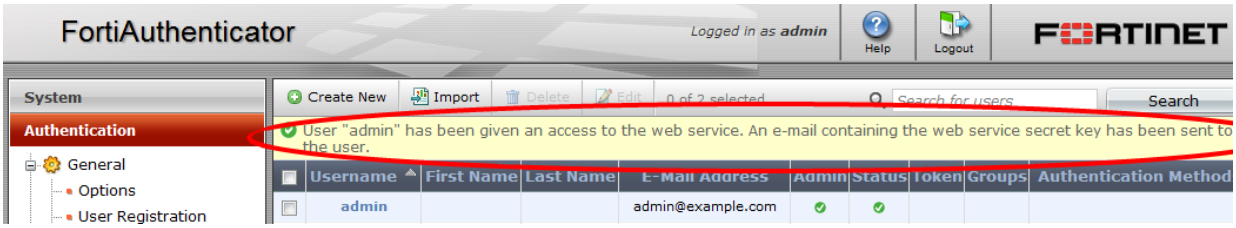
The option *Allow RADIUS authentication* is removed when elevating a user to an FortiAuthenticator administrator.

Create a new user or edit an existing one. In the example shown, the admin account is used.

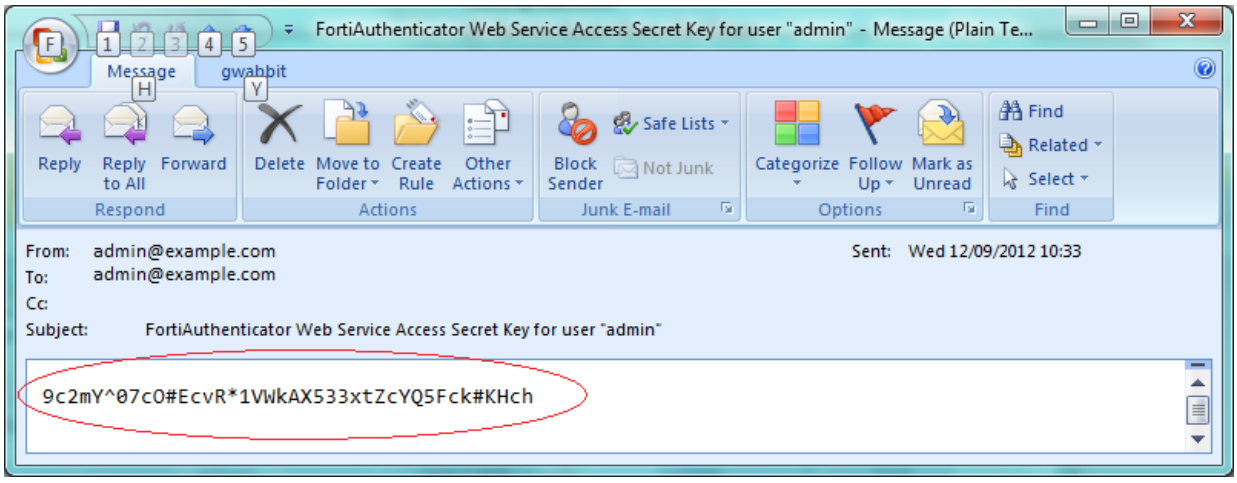
- Select *User Role: Administrator*
- Enable *Web Service Access*
- Enter a valid email address. The API Key will be forwarded to this address so ensure it is valid and email routing is working beforehand.



- Click *OK* to save the details.
- The API Web Service Access Key used to authenticate to the API will be e-mailed to the API administrator.



- Make a note of the API Web Service Access Key



Should the API Web Service Access Key be lost, access can be recovered by disabling the Web Service feature for the user, saving and then re-enabling the feature. A new key will be generated (and all code using it will need to be updated with the new credentials).

Accessing the REST API

The FortiAuthenticator API can be accessed from most browsers (GET) however browser add-ons may be required for extended operations (e.g. PUT). More complicated, scripted queries can be made using utilities such as cURL and most scripting languages such as Perl or Python have built in libraries for interacting with RESTful APIs.

Example shown within this document will be demonstrated with the cross platform utility cURL.

All of the resource URLs are in this form:

https://[server_name]/api/[api_version]/[resource]/

where:

`server_name`	=	Name or IP of the FortiAuthenticator
`api_version`	=	API version to be used (currently v1)
`resource`	=	Resource or part of config to be viewed
`id`	=	Resource ID to view, edit, or delete

Filtering query results

Queries to the API can be to modify the query/response format or to filter the results. Below are some arguments that can be passed to the REST API URL. Please refer to the specific resource documentation to find

out which of these filter operations are allowed.

`?format=[format_type]`	where format _type= xml or json (default)
`?limit=[integer]`	where integer specifies number of records to return (default unlimited)
`?offset=[integer]`	where integer specifies number of items in resource list to skip e.g. if there are 10 items, to return item #5 - #10 only, specify offset=4
`?order_by=[field]`	order returned list by a known field name (e.g.?order_by=name)

Field filters

- exact: search for an exact match
  (e.g. to return items that has a name matching "John Doe", `name__exact=John Doe`)
- in: search for items that matches specific filter criteria
  (e.g. to return items that has a name matching "John" or "Bill", `?name__in=John&name__in=Bill`)

Supported API Methods

All of the resource URLs are in this form: https://[server_name]/api/[api_version]/[resource]/. The current API version is v1.

To list all of the available resource end-points, send a request to:

https://[server_name]/api/v1/?format=xml

To view schema, supported methods and available fields for each end-point, append /schema/ to the end-point URL. For example, to view schema for /auth/ API, perform a GET request to:

https://[server_name]/api/v1/auth/schema/?format=xml

In general, an end-point may support the following methods, though not all methods are supported by all end-points (see each end-point's documentation for the list of allowed methods):

Method	URL	Operation Description	Success Response Code
`GET (list)`	`/[resource]/`	Retrieve a list of all resources for the endpoint	200 OK
`GET (detail)`	`/[resource]/[id]/`	Retrieve a specific resource with ID id from the endpoint	200 OK

Method	URL	Operation Description	Success Response Code
POST	/[resource]/	Create a new resource on the given endpoint. The data being POST-ed must follow the same format as the data returned by the GET parameter	201 CREATED
PUT (list)	/[resource]/	Update all of the resources for the given endpoint. Any existing items will be replaced with the new data. Data must follow the same format as the data returned by the GET parameter.	204 NO CONTENT
PUT (detail)	/[resource]/[id]/	Update an existing item specified with ID id. Data must follow the same format as the data returned by the GET parameter.	204 NO CONTENT
PATCH (detail)	/[resource]/[id]/	Update specific fields on an existing item with ID id	202 ACCEPTED
DELETE (list)	/[resource]/	Delete all resources from an endpoint	204 NO CONTENT
DELETE (detail)	/[resource]/[id]/	Delete an existing resource specified with ID id from an endpoint	204 NO CONTENT

Supported Data Formats

Currently, JSON and XML are supported. To specify a format on the request:

For a GET request, there are 2 options:

- Use the GET format parameter (e.g. ?format=json or ?format=xml)
- Specify an Accept HTTP header with a correct mimetype (e.g. Accepts: application/json for JSON)

The GET format parameter takes precedence over the Accept header.

Browsers will usually default to requesting for an XML data type when format is not specified for a GET request.

Resource Summary

There are currently 8 main resources and the root record which can be accessed via the API:

Resource	URL	Operation Description	Supported Methods
Root	/	Allows querying of available resources.	GET
Local User Management	/localusers/	Allows the creation, modification and deletion of user accounts.	GET, POST, PATCH
Local Group Management	/usergroups/	Allows the creation and deletion of user groups and specify users within that group.	GET, POST, PUT, DELETE
Local Group Membership	/localgroup-memberships/	Represents local user group membership resource (relationship between local user and local user group).	GET, POST, DELETE
User Authentication	/auth/	Allows validation of user authentication credentials.	POST
FortiToken	/fortitokens/	Allows provisioning of FortiTokens.	
SSO Group	/ssogroup/	Enables remote configuration of the SSO & Dynamic Policies → SSO → SSO Groups table.	GET, POST, DELETE
FortiGate Filter Group	/fgtgroupfilter/	Enables remote configuration of the SSO & Dynamic Policies → SSO→ FortiGate Group Filtering table.	GET, PUT
SSO Authentication	/ssoauth/	Adds/removes a user from the FSSO logged in users table.	POST

Example API Calls

For the purpose of these examples, cURL is being used to make the requests. cURL is more flexible than a browser alone, is cross platform and can be called from most scripts. It is not as flexible as native scripting languages but is a good clear example which can be used to understand how the API functions.

The following flags are used in the cURL query:

- -kIgnore certificate errors. This can be overcome with use of a valid certificate.
- -vVerbose. Increase the level of logging information (useful for debugging).
- -uUser login information in format USER[:PASSWORD]

When using PUT/POST with cURL on Windows, problems can be encountered with escaping of the required double quotes in the data content, leading to errors related to incomplete closed brackets. To avoid this, the code should be properly escaped (using \ before any double quotes) or the data text stored in a file and referenced using:

```
-d @<filename>
```

Alternatively, it is highly recommended that this is run on a Linux OS, where escaping of characters in cURL is more predictable.

General API Usage

View Available Endpoint Resources

JSON Query

- JSON specified via GET

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    https://192.168.0.122/api/v1/?format=json
```

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"  -H 'Accept:
    application/json' https://192.168.0.122/api/v1/
```

Response

```
< HTTP/1.1 200 OK< Date: Mon, 09 Jun 2014 10:51:23 GMT< Server: Apache< Vary:
    Accept-Language,Cookie< X-Frame-Options: SAMEORIGIN< Content-Language: en<
    Transfer-Encoding: chunked< Content-Type: application/json<* Connection #0
    to host 192.168.0.122 left intact* Closing connection #0

{"auth": {"list_endpoint": "/api/v1/auth/", "schema": "/api/v1/auth/schema/"},
    "fgtgroupfilter": {"list_endpoint": "/api/v1/fgtgroupfilter/", "schema":
    "/api/v1/fgtgroupfilter/schema/"}, "fortitokens": {"list_endpoint":
    "/api/v1/fortitokens/", "schema": "/api/v1/fortitokens/schema/"}, "localusers":
    {"list_endpoint": "/api/v1/localusers/", "schema": "/api/v1/localusers/schema/"},
    "ssoauth": {"list_endpoint": "/api/v1/ssoauth/", "schema": "/api/v1/ssoauth/schema/"},
```

```
        "ssogroup": {"list_endpoint": "/api/v1/ssogroup/", "schema":
        "/api/v1/ssogroup/schema/"}, "usergroups": {"list_endpoint": "/api/v1/usergroups/",
        "schema": "/api/v1/usergroups/schema/"}}
```

XML Query

- XML specified via GET

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    https://192.168.0.122/api/v1/?format=xml
```

- XML specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -H 'Accept:
    application/xml' https://192.168.0.122/api/v1/
```

Response

```
< HTTP/1.1 200 OK
< Date: Mon, 09 Jun 2014 11:03:25 GMT
< Server: Apache
< Vary: Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Transfer-Encoding: chunked
< Content-Type: application/xml; charset=utf-8
<
<?xml version='1.0' encoding='utf-8'?>
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
<response><fgtgroupfilter type="hash"><list_endpoint>/api/v1/fgtgroupfilter/</list_
    endpoint><schema>/api/v1/fgtgroupfilter/schema/</schema></fgtgroupfilter><localusers
    type="hash"><list_endpoint>/api/v1/localusers/</list_
    endpoint><schema>/api/v1/localusers/schema/</schema></localusers><usergroups
    type="hash"><list_endpoint>/api/v1/usergroups/</list_
    endpoint><schema>/api/v1/usergroups/schema/</schema></usergroups><auth
    type="hash"><list_endpoint>/api/v1/auth/</list_
    endpoint><schema>/api/v1/auth/schema/</schema></auth><fortitokens type="hash"><list_
    endpoint>/api/v1/fortitokens/</list_
    endpoint><schema>/api/v1/fortitokens/schema/</schema></fortitokens><ssogroup
    type="hash"><list_endpoint>/api/v1/ssogroup/</list_
    endpoint><schema>/api/v1/ssogroup/schema/</schema></ssogroup><ssoauth
    type="hash"><list_endpoint>/api/v1/ssoauth/</list_
    endpoint><schema>/api/v1/ssoauth/schema/</schema></ssoauth></response>
```

User Groups (/usergroups/)

URL: `https://[server_name]/api/[api_version]/usergroups/`

This end-point represents the user group resource. In the FortiAuthenticator GUI, this resource corresponds to Authentication → User Groups. This API is for use by third party user provisioning systems.

Supported Fields

Field	Description	Type	Required	Other Restriction
name	Group name	String	Yes	max length = 50
users	List of local users in the group	List	No	List of local users URI

Allowed Methods

Allowed Methods	Resource URI	Action
GET		Get all groups and associated users
POST		Create a new user
PUT		Replaces all of the resources for the group. This is done by removing all existing items first before creating the new items. Data must follow the same format as the data returned by the GET parameter.
PATCH		Add users to a user group
DELETE		Delete a specified group

Allowed Filters

Field	Filters
name	`exact`

View All User Groups

JSON Query

- JSON specified via GET

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    https://192.168.0.122/api/v1/usergroups/?format=xml
```

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -H 'Accept:
    application/xml' https://192.168.0.122/api/v1/usergroups/
```

Response

```
< HTTP/1.1 200 OK
< Date: Mon, 09 Jun 2014 11:46:34 GMT
< Server: Apache
```

```
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/xml; charset=utf-8
<
<?xml version='1.0' encoding='utf-8'?>
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
<response>

<objects type="list"><object><users type="list"/>
<idtype="integer">5</id><name>REST_RADIUS</name><resource_
    uri>/api/v1/usergroups/5/</resource_uri></object>

<object><users type="list"/>
<idtype="integer">4</id><name>Test_LDAP</name><resource_
    uri>/api/v1/usergroups/4/</resource_uri></object>

<object><users type="list"><value>/api/v1/localusers/4/</value></users>
<idtype="integer">3</id><name>Test_Local</name><resource_
    uri>/api/v1/usergroups/3/</resource_uri></object></objects>

<meta type="hash"><next type="null"/><total_count type="integer">3</total_count><previous
    type="null"/><limit type="integer">20</limit><offset
    type="integer">0</offset></meta></response>
```

The response above has been reformatted with carriage returns to make the results more clear.

The response shows that there are 3 groups already configured (in RED).

- Test_RADIUS (in ID position 5)
- Test_LDAP (in ID position 4)
- Test_Local (in ID position 3)

Test_RADIUS and Test_LDAP groups do not contain any users, however, the Test_Local group contains 1 user, identified as local user with ID=4 (in GREEN). See the LocalUsers for identifying Usernames from user IDs.

The total number of configured and supported User Groups is also returned for troubleshooting purposes (in GOLD).

Create a User Group

JSON Query

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -X POST -d '
    {"name":"Group999"}' -H 'Content-Type: application/json'
    https://192.168.0.122/api/v1/usergroups/
```

Response

```
< HTTP/1.1 201 CREATED
< Date: Mon, 09 Jun 2014 12:02:33 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Location: https://192.168.0.122/api/v1/usergroups/6/
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
```

Verify user group creation

Use API call documented in "Local Users (/localusers/)" on page 28

Field	Lookup Expressions	Values
username	exact, iexact, contains, icontains, in	
first_name	exact, iexact, contains, icontains	
last_name	exact, iexact, contains, icontains	
email	exact, iexact, contains, icontains, in	
active	exact	
city	exact, iexact, contains, icontains	
state	exact, iexact, contains, icontains	
country	exact, iexact, contains, icontains	
token_type		ftk, ftm, email, sms
token_serial	exact, iexact	

Third-party Integration: FTM Provisioning

For integration with a third-party authentication server which needs to manage token validation, it is possible for the FortiAuthenticator to return FTM seed during provisioning. However, certain conditions must be met:

- Seed may only be returned when creating a new local user via POST method and when provisioning an FTM to an existing user via PATCH method
- A GET URL parameter (returnseed=1) needs to be specified to explicitly tell FAC to return an encrypted seed for the token (e.g. https://[server_name]/api/v1/localusers/2/?returnseed=1)
- A seed encryption passphrase must be specified in FortiGuard settings.

The seed is encrypted and returned as a PSKC XML file string according to RFC 6030. The key is derived from the configured passphrase using the PBKDF2 key derivation function (32 byte key length, 1000 iterations), encrypted with AES 256 CBC encryption, and signed with a SHA256 HMAC.

Whenever a FortiToken Mobile is provisioned, its activation code will be returned as well.

List All Local Users above

```
< HTTP/1.1 200 OK
< Date: Mon, 09 Jun 2014 12:18:19 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/xml; charset=utf-8
<
<?xml version='1.0' encoding='utf-8'?>
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
<response><objects type="list"><object><users type="list"/><id type="integer">6</id>
    <name>Group999</name><resource_uri>/api/v1/usergroups/6/</resource_
    uri></object><object><users type="list"/><id type="integer">5</id><name>REST_
    RADIUS</name><resource_uri>/api/v1/usergroups/5/</resource_uri></object><object><users
    type="list"/><id type="integer">4</id><name>Test_LDAP</name><resource_
    uri>/api/v1/usergroups/4/</resource_uri></object><object><users
    type="list"><value>/api/v1/localusers/4/</value></users><id
    type="integer">3</id><name>Test_Local</name><resource_
    uri>/api/v1/usergroups/3/</resource_uri></object></objects><meta type="hash"><next
    type="null"/><total_count type="integer">4</total_count><previous type="null"/><limit
    type="integer">20</limit><offset type="integer">0</offset></meta></response>
```

Attempt to create a user Group with the same name

```
< HTTP/1.1 400 BAD REQUEST
< Date: Mon, 09 Jun 2014 12:04:06 GMT
< Server: Apache
< Vary: Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Connection: close
< Transfer-Encoding: chunked
< Content-Type: application/json
<
* Closing connection #0
{"usergroups": {"name": ["A user group with that name already exists."]}}
```

Add a user to a group

Note, the required users should be elucidated by querying the /localusers/ list as documented in the "Local Users (/localusers/)" on page 26 section. In this example:

test_user	=	/api/v1/localusers/5/
test_user2	=	/api/v1/localusers/5/

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -X PATCH -d '{"users":
    ["/api/v1/localusers/5/","/api/v1/localusers/4/"]}' -H 'Content-Type:
    application/json' https://192.168.0.122/api/v1/usergroups/9/
```

This command is not additive i.e. adding a single user entry will not increment the list it will overwrite. Using {"users":[ ]} for example will clear the users list.

Delete a User Group

JSON Query

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -X DELETE -H 'Content-Type:
    application/json' https://192.168.0.122/api/v1/usergroups/6/
```

Response

```
< HTTP/1.1 204 NO CONTENT
< Date: Mon, 09 Jun 2014 12:25:18 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
```

Note that 204 NO CONTENT shows that the group has been successfully deleted. A subsequent listing confirms this as Group999 no longer exists:

```
< HTTP/1.1 200 OK
< Date: Mon, 09 Jun 2014 12:26:05 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/xml; charset=utf-8
<
<?xml version='1.0' encoding='utf-8'?>
```

```
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
<response><objects type="list"><object><users type="list"/><id
    type="integer">5</id><name>REST_RADIUS</name><resource_
    uri>/api/v1/usergroups/5/</resource_uri></object><object><users type="list"/><id
    type="integer">4</id><name>Test_LDAP</name><resource_
    uri>/api/v1/usergroups/4/</resource_uri></object><object><users
    type="list"><value>/api/v1/localusers/4/</value></users><id
    type="integer">3</id><name>Test_Local</name><resource_
    uri>/api/v1/usergroups/3/</resource_uri></object></objects><meta type="hash"><next
    type="null"/><total_count type="integer">3</total_count><previous type="null"/><limit
    type="integer">20</limit><offset type="integer">0</offset></meta></response>
    [Carl@CentOS ~]$
```

The Delete command will delete the group even if the group contains users or if it is in use e.g. in a RADIUS Client configuration. Checks should be made prior to executing this command.

View a specific User Group

JSON Query

- JSON specified via GET

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    "https://192.168.0.122/api/v1/usergroups/?format=json&name=/api/v1/usergroups/8/"
```

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -H 'Accept:
    application/json'"https://192.168.0.122/api/v1/usergroups/?format=json&name=Group999"
```

The filter used in this situation is the group “name” not the URL or ID.

The URL requires additional quoting in this case otherwise the Unix CLI treats the “&” as an instruction to place the cURL command into the background.

Querying a non-existent group will return a successful 200 OK response with empty object data. This is by design as this is not necessarily an error situation.

Response

```
< HTTP/1.1 200 OK
< Date: Tue, 10 Jun 2014 10:11:47 GMT
```

```
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
{"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 1},
    "objects": [{"id": 9, "name": "Group999", "resource_uri": "/api/v1/usergroups/9/",
    "users": ["/api/v1/localusers/5/"]}]}
```

FortiTokens (/fortitoken/)

URL: `https://[server_name]/api/[api_version]/fortitokens/`

This end-point represents the FortiToken resource. In the FortiAuthenticator GUI, this resource corresponds to User Management → FortiTokens. This API is for use by third party user provisioning systems to ascertain which tokens are available to be provisioned to a user.

Supported Fields

Field	Display Name	Type	Required	Other Restriction
serial	Serial number	string	No	
type	Type	string	No	Either `ftk` or `ftm`
status	Status	string	No	One of `new, available, pending, assigned`

Allowed Methods

Type	Allowed Methods	Action
List	GET	Get all FortiTokens

Allowed Filters

Field	Lookup Expressions	Values
serial	exact, iexact	
type	exact, iexact	`ftk, ftm`
status	exact, iexact	`new, available, pending, assigned`

View All Tokens

JSON Query

- JSON specified via GET

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    https://192.168.0.122/api/v1/fortitokens/?format=json
```

Response

```
< HTTP/1.1 200 OK
< Date: Mon, 09 Jun 2014 18:17:42 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
{"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 2},
    "objects": [{"resource_uri": "/api/v1/fortitokens/1/", "serial": "FTKMOB44142CCBF3",
    "status": "available", "type": "ftm"}, {"resource_uri": "/api/v1/fortitokens/2/",
    "serial": "FTKMOB4471BB94D1", "status": "available", "type": "ftm"}]}
```

View subset of tokens using filters

This example shows how it is possible to obtain a list of specific tokens e.g. The first available FortiToken Mobile token.

JSON Query

- JSON specified via GET

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -H 'Accept:
    application/json'
    "https://192.168.0.122/api/v1/fortitokens/?format=json&type=ftm&status=available&limit
    =1"
```

The URL requires additional quoting in this case otherwise the Unix CLI treats the “&” as an instruction to place the cURL command into the background.

Response

```
< HTTP/1.1 200 OK
< Date: Mon, 09 Jun 2014 18:17:42 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
{"meta": {"limit": 1, "next":
    "/api/v1/fortitokens/?status=available&type=ftm&offset=1&limit=1&format=json",
    "offset": 0, "previous": null, "total_count": 2}, "objects": [{"resource_uri":
    "/api/v1/fortitokens/1/", "serial": "FTKMOB44142CCBF3", "status": "available", "type":
    "ftm"}]}
```

Local Users (/localusers/)

URL: `https://[server_name]/api/[api_version]/localusers/`

This end-point represents local user resource i.e. a user account. In the FortiAuthenticator GUI, this resource corresponds to Authentication → Local Users. This API is for use by third party provisioning systems.

Supported Fields

Field	Display Name	Type	Required	Other Restriction
address	Address	string	No	max length = 80
city	City	string	No	max length = 40
country	Country	string	No	Must be a country code from ISO-3166 list
custom1	Custom user field 1	string	No	max length = 255
custom2	Custom user field 2	string	No	max length = 255
custom3	Custom user field 3	string	No	max length = 255
email	E-mail address	string	No	Must be a valid e-mail address
first_ name	First name	string	No	max length = 30
last_ name	Last name	string	No	max length = 30
mobile_ number	Mobile number	string	No	max length = 25, must follow international number format: +[country_code]-[number]
phone_ number	Mobile number	string	No	max length = 25
state	State or province	string	No	max length = 40

Field	Display Name	Type	Required	Other Restriction
user_groups	Local user groups that this user is a member of	list	No	List of user groups URI
token_auth*	Token Auth	boolean	No	Whether second factor authentication should be enabled. If 'true', token_type is required.
token_type*	Token Type	string	No	One of ftk, ftm, email, sms. If email is chosen, email is required. If sms is chosen, mobile_number is required.
token_serial*	Token Serial	string	No	If token_type is ftm, or ftk, and this is not present or blank, the next available token will be assigned.
ftk_only	Enable FortiToken-only authentication	boolean	No	If set, token_auth must be true, and token_type must be either ftk or ftm. If this field is changed to false, email must be set to reset user's password and send a new random password. Mutually exclusive with password.
expires_at	Expiration time	string	No	ISO-8601 formatted user expiration time in UTC. Specified time should be formatted using ISO-8601 with a timezone offset. If timezone info is not set, time is always assumed to be in UTC. To remove an expiration time, set this field to an empty string. Time must be at least an hour in the future.

Additionally, when creating a new user, the following two fields are available.

Field	Display Name	Type	Required	Other Restriction
password	Password	string	No	max length = 50

Allowed Methods

HTTP Method	Resource URI	Action
GET	/api/v1/localusers/	Get all regular local users
GET	/api/v1/localusers/[id]/	Get a specific local user with ID id

HTTP Method	Resource URI	Action
POST	/api/v1/localusers/	Create a new local user **Notes:** • If password is specified, that password will be set. • If password is not specified, email field becomes required, and a random password will be created and e-mailed to the new user.
POST	/api/v1/localusers/[id]/sendoobtoken/	Send an out-of-band token code (email/SMS token) to a local user
PATCH	/api/v1/localusers/[id]/	Update specified fields for a specific local user with ID id
DELETE	/api/v1/localusers/[id]/	Delete a local user

Allowed Filters

Field	Lookup Expressions	Values
username	`exact, iexact, contains, icontains, in`	
first_name	`exact, iexact, contains, icontains`	
last_name	`exact, iexact, contains, icontains`	
email	`exact, iexact, contains, icontains, in`	
active	`exact`	
city	`exact, iexact, contains, icontains`	
state	`exact, iexact, contains, icontains`	
country	`exact, iexact, contains, icontains`	
token_type		`ftk, ftm, email, sms`
token_serial	`exact, iexact`	

Third-party Integration: FTM Provisioning

For integration with a third-party authentication server which needs to manage token validation, it is possible for the FortiAuthenticator to return FTM seed during provisioning. However, certain conditions must be met:

- Seed may only be returned when creating a new local user via POST method and when provisioning an FTM to an existing user via PATCH method

- A GET URL parameter (returnseed=1) needs to be specified to explicitly tell FAC to return an encrypted seed for the token (e.g. https://[server_name]/api/v1/localusers/2/?returnseed=1)
- A seed encryption passphrase must be specified in FortiGuard settings.

The seed is encrypted and returned as a PSKC XML file string according to RFC 6030. The key is derived from the configured passphrase using the PBKDF2 key derivation function (32 byte key length, 1000 iterations), encrypted with AES 256 CBC encryption, and signed with a SHA256 HMAC.

Whenever a FortiToken Mobile is provisioned, its activation code will be returned as well.

List All Local Users

JSON Query

- JSON specified via GET

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    https://192.168.0.122/api/v1/localusers/?format=xml
```

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -H 'Accept:
    application/xml' https://192.168.0.122/api/v1/ localusers/
```

Response

```
< HTTP/1.1 200 OK
< Date: Mon, 09 Jun 2014 20:14:23 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
{"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 2},
    "objects": [{"address": "", "city": "", "country": "", "custom1": "", "custom2": "",
    "custom3": "", "email": "", "first_name": "", "id": 5, "last_name": "", "mobile_
    number": "", "phone_number": "", "resource_uri": "/api/v1/localusers/5/", "state": "",
    "token_auth": false, "token_serial": "", "token_type": null, "user_groups":
    ["/api/v1/usergroups/9/", "/api/v1/usergroups/8/"], "username": "test_user2"},
    {"address": "", "city": "", "country": "", "custom1": "", "custom2": "", "custom3":
    "", "email": "", "first_name": "", "id": 4, "last_name": "", "mobile_number": "",
    "phone_number": "", "resource_uri": "/api/v1/localusers/4/", "state": "", "token_
    auth": false, "token_serial": "", "token_type": null, "user_groups":
    ["/api/v1/usergroups/8/"], "username": "test_user"}]}
```

Here you will notice that there are 2 users defined “test_user” and “test_user2”. Note that admin users are not returned by the localusers query.

Create Local User

JSON Query

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -X POST -d '
    {"username":"test_user3","password":"testpassword","email":"test_
    user3@example.com","mobile":"+44-1234567890"}' -H 'Content-Type: application/json'
    https://192.168.0.122/api/v1/localusers/
```

Response

```
< HTTP/1.1 201 CREATED
< Date: Mon, 09 Jun 2014 20:29:20 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Location: https://192.168.0.122/api/v1/localusers/6/
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
```

Verify user creation

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    https://192.168.0.122/api/v1/localusers/?format=json


< HTTP/1.1 200 OK
< Date: Mon, 09 Jun 2014 20:30:26 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
{"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 3},
    "objects": [{"address": "", "city": "", "country": "", "custom1": "", "custom2": "",
    "custom3": "", "email": "", "first_name": "", "id": 5, "last_name": "", "mobile_
    number": "", "phone_number": "", "resource_uri": "/api/v1/localusers/5/", "state": "",
    "token_auth": false, "token_serial": "", "token_type": null, "user_groups":
    ["/api/v1/usergroups/9/", "/api/v1/usergroups/8/"], "username": "test_user2"},
    {"address": "", "city": "", "country": "", "custom1": "", "custom2": "", "custom3":
    "", "email": "", "first_name": "", "id": 4, "last_name": "", "mobile_number": "",
    "phone_number": "", "resource_uri": "/api/v1/localusers/4/", "state": "", "token_
    auth": false, "token_serial": "", "token_type": null, "user_groups":
    ["/api/v1/usergroups/8/"], "username": "test_user"}, {"address": "", "city": "",
    "country": "", "custom1": "", "custom2": "", "custom3": "", "email": "test_
    user3@example.com", "first_name": "", "id": 6, "last_name": "", "mobile_number": "",
    "phone_number": "", "resource_uri": "/api/v1/localusers/6/", "state": "", "token_
    auth": false, "token_serial": "", "token_type": null, "user_groups": [], "username":
    "test_user3"}]}
```

Modify Local User

JSON Query

- JSON specified via Accept Header

  Modify the newly created user “test_user3” aka User ID == 6 using the PATCH command.

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -X PATCH -d '
    {"custom1":"example","country":"GB"}' -H 'Content-Type: application/json'
    https://192.168.0.122/api/v1/localusers/6/
```

Response

```
< HTTP/1.1 202 ACCEPTED
< Date: Mon, 09 Jun 2014 21:07:28 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
```

Delete Local User

Deleing a user via the API is currently not supported.

Applying Filters

List Specific Local User

JSON Query

- JSON specified via GET

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    "https://192.168.0.122/api/v1/localusers/?format=json&username=test_user3"
```

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -H 'Accept:
    application/json' "https://192.168.0.122/api/v1/localusers/?username=test_user3"
```

Response

```
< HTTP/1.1 200 OK
< Date: Tue, 10 Jun 2014 11:06:20 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
{"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 1},
    "objects": [{"address": "", "city": "", "country": "", "custom1": "example",
    "custom2": "", "custom3": "", "email": "test_user3@example.com", "first_name": "",
    "id": 6, "last_name": "", "mobile_number": "", "phone_number": "", "resource_uri":
    "/api/v1/localusers/6/", "state": "", "token_auth": false, "token_serial": "", "token_
    type": null, "user_groups": [], "username": "test_user3"}]}[
```

View all users from Country=GB

JSON Query

- JSON specified via Accept Header

  View all users from the country GB (Great Britain).

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -H 'Accept:
    application/json' "https://192.168.0.122/api/v1/localusers/?country=GB"
```

Response

```
< HTTP/1.1 200 OK
< Date: Tue, 10 Jun 2014 11:14:39 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
{"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 2},
    "objects": [{"address": "", "city": "", "country": "GB", "custom1": "example",
    "custom2": "", "custom3": "", "email": "test_user3@example.com", "first_name": "",
    "id": 6, "last_name": "", "mobile_number": "", "phone_number": "", "resource_uri":
    "/api/v1/localusers/6/", "state": "", "token_auth": false, "token_serial": "", "token_
    type": null, "user_groups": [], "username": "test_user3"}, {"address": "", "city": "",
    "country": "GB", "custom1": "example", "custom2": "", "custom3": "", "email": "",
    "first_name": "", "id": 5, "last_name": "", "mobile_number": "", "phone_number": "",
    "resource_uri": "/api/v1/localusers/5/", "state": "", "token_auth": false, "token_
    serial": "", "token_type": null, "user_groups": ["/api/v1/usergroups/9/",
    "/api/v1/usergroups/8/"], "username": "test_user2"}]}[
```

LDAP Users (/ldapusers/)

URL: `https://[server_name]/api/[api_version]/ldapusers/`

This end-point represents imported remote LDAP user resource. In the FortiAuthenticator GUI, this resource corresponds to Authentication → Remote Auth Servers → LDAP.

Supported Fields

Field	Display Name	Type	Required	Other Restriction
username	Username	string	Yes	Read-only
dn	Distinguished name	string	Yes	Read-only

Field	Display Name	Type	Required	Other Restriction
server_ name	Server name	string	No	Read-only
server_ address	Server address	string	No	Read-only
email	E-mail address	string	No	Must be a valid e-mail address
first_ name	First name	string	No	max length = 30
last_ name	Last name	string	No	max length = 30
active	Account Status	boolean	No	
mobile_ number	Mobile number	string	No	max length = 25, must follow international number format: +[country_code]-[number]
token_ auth	Token Auth	boolean	No	Whether second factor authentication should be enabled. If true, token_type is required.
token_ type	Token Type	string	No	One of ftk, ftm, email, sms. If email is chosen, email is required. If SMS is chosen, mobile_number is required.
token_ serial	Token Serial	string	No	If token_type is ftm, or ftk, and this is not present or blank, the next available token will be assigned.

Allowed Methods

HTTP Method	Resource URI	Action
GET	/api/v1/ldapusers/	Get all non-admin LDAP users
GET	/api/v1/ldapusers/[id]/	Get a specific non-admin LDAP user
POST	/api/v1/ldapusers/[id] /sendoobtoken/	Send an out-of-band token code (email/SMS token) to an LDAP user
POST	/api/v1/localusers/[id] /sendoobtoken/	Send an out-of-band token code (email/SMS token) to a local user
PATCH	/api/v1/localusers/[id]/	Update specified fields for a specific local user with ID id

Allowed Filters

Field	Lookup Expressions	Values
username	exact, iexact, contains, icontains, in	
dn	exact, iexact, contains, icontains	
first_name	exact, iexact, contains, icontains, in	
last_name	exact, iexact, contains, icontains, in	
email	exact, iexact, contains, icontains, in	
active	exact	
server_name	exact, iexact, contains, icontains	
server_address	exact, iexact, contains, icontains	
token_type		ftk, ftm, email, sms
token_serial	exact, iexact	

Third-party Integration: FTM Provisioning

For integration with a third-party authentication server which needs to manage token validation, it is possible for the FortiAuthenticator to return FTM seed during provisioning. However, certain conditions must be met:

- Seed may only be returned when provisioning an FTM to an existing user via PATCH method
- A GET URL parameter (returnseed=1) needs to be specified to explicitly tell FAC to return an encrypted seed for the token (e.g. https://[server_name]/api/v1/ldapusers/2/?returnseed=1)
- A seed encryption passphrase must be specified in FortiGuard settings.

The seed is encrypted and returned as a PSKC XML file string according to RFC 6030. The key is derived from the configured passphrase using the PBKDF2 key derivation function (32 byte key length, 1000 iterations), encrypted with AES 256 CBC encryption, and signed with a SHA256 HMAC.

Whenever a FortiToken Mobile is provisioned, its activation code will be returned as well.

Local User Group Memberships (/localgroup-memberships/)

URL: `https://[server_name]/api/[api_version]/localgroup-memberships/`

This end-point represents local user group membership resource (relationship between local user and local user group).

Support Fields

Field	Description	Type	Required	Read-only	Other Restriction
group	Group	string	Yes		Local user group URI
user	Member of the group	string	Yes		Local user URI
group_name	Member of the group	string	No	Yes	
username	Member username	string	No	Yes	

Allowed Methods

HTTP Method	Resource URI	Action
GET	/api/v1/localgroup-memberships/	Get all local group memberships
GET	/api/v1/localgroup-memberships/[id][	Get a specific local group membership
POST	/api/v1/localgroup-memberships/	Create a new local group membership
DELETE	/api/v1/localgroup-memberships/[id]	Delete a local group membership

Allowed Filters

Field	Filters	Description
group	exact, in	Accepts group ID (e.g. group=15)
user	exact, in	Accepts user ID
group_name	exact, iexact, contains, icontains, in	
username	exact, iexact, contains, icontains, in	

SSO/Remote Groups (/ssogroup/)

URL: `https://[server_name]/api/[api_version]/ssogroup/`

In the FortiAuthenticator GUI, this resource corresponds to Fortinet SSO Methods → SSO → SSO Groups.

Supported Fields

Field	Display Name	Type	Required	Other Restriction
name	Name	string	Yes	max length=50, unique

Allowed Methods

HTTP Method	Resource URI	Action
GET	/api/v1/ssogroup/	Get all SSO groups
GET	/api/v1/ssogroup/[id]/	Get an SSO group with ID id
POST	/api/v1/ssogroup/	Create a new SSO group
DELETE	/api/v1/ssogroup/	Delete all SSO groups
DELETE	/api/v1/ssogroup/[id]/	Delete an SSO group with ID id

Allowed Filters

Field	Lookup Expressions
name	exact, in

View SSO Group Configuration

JSON Query

- JSON specified via GET

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    https://192.168.0.122/api/v1/ssogroup/?format=json
```

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -H 'Accept:
    application/json' https://192.168.0.122/api/v1/ssogroup/
```

Response

```
< HTTP/1.1 200 OK
< Date: Tue, 10 Jun 2014 11:48:08 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
{"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 1},
    "objects": [{"id": 1, "name": "Test_Group1", "resource_uri": "/api/v1/ssogroup/1/"}]}
    {"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 1},
    "objects": [{"id": 1, "name": "Test_Group1", "resource_uri": "/api/v1/ssogroup/1/"}]}
```

JSON Query

- JSON specified via GET

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    https://192.168.0.122/api/v1/ssogroup/?format=json
```

- JSON specified via Accept Header

```
curl -k -v -u "zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -H 'Accept: application/json'
    https://192.168.0.122/api/v1/ssogroup/
```

Response

```
< HTTP/1.1 200 OK
< Date: Tue, 10 Jun 2014 11:48:08 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
{"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 1},
    "objects": [{"id": 1, "name": "Test_Group1", "resource_uri": "/api/v1/ssogroup/1/"}]}
    {"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 1},
    "objects": [{"id": 1, "name": "Test_Group1", "resource_uri": "/api/v1/ssogroup/1/"}]}
```

Create SSO Group

JSON Query

- JSON specified via POST

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -X POST -d '{"name":"Test_
    Group2"}' -H 'Content-Type: application/json' https://192.168.0.122/api/v1/ssogroup/
```

XML Query

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -X POST -d
    '<object><name>Test_Group2</name></object>' -H 'Content-Type: application/xml'
    https://192.168.0.122/api/v1/ssogroup/
```

Response

```
< HTTP/1.1 201 CREATED
< Date: Tue, 10 Jun 2014 11:51:31 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Location: https://192.168.0.122/api/v1/ssogroup/3/
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
```

Successful 201 CREATED response code. See "Appendix A –API Response Codes" on page 53 for full details.

Filter Lookup Expressions

Expression	Description
`exact`	search for an exact match (e.g. name`__exact=John Doe`, would return user with name "John Doe", but not "john doe")
`iexact`	search for a case-insensitive exact match (e.g. name__iexact=john doe, would return user with name "John Doe")
`contains`	search for an item that contains a specific keyword
`icontains`	same as above, but case-insensitive
`in`	search for items that matches specific filter criteria (e.g. to return items that has a name matching "John" or "Bill", ?name`__in=John&name__in=Bill`)
`startswith`	search for items that starts with a text
`istartswith`	same as above, but case-insensitive

See Appendix A –API Response Codes on page 53 for full details.

Delete SSO Group

Query

- Specified via POST

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -X DELETE
    https://192.168.0.122/api/v1/ssogroup/3/
```

Response

```
< HTTP/1.1 204 NO CONTENT
< Date: Tue, 10 Jun 2014 11:53:52 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
```

204 NO CONTENT is a successful result. Verify by querying /ssogroup/ to verify group 3 has been deleted.

FortiGate Group Filter (/fgtgroupfilter/)

URL: `https://[server_name]/api/[api_version]/fgtgroupfilter/`

In the FortiAuthenticator GUI, this resource corresponds to *Fortinet SSO Methods → SSO → FortiGate Group Filtering*.

Supported Fields

Field	Display Name	Type	Required	Other Restriction
shortname	Name	string	Yes	max length=32, unique
nasname	NAS name/IP	string	Yes	max length=128, unique

Allowed Methods

HTTP Method	Resource URI	Action
GET	/api/v1/fgtgroupfilter/	Get all FortiGate Group Filters
GET	/api/v1/fgtgroupfilter/[id]/	Get a specific FortiGate Group Filter with ID id
PUT	/api/v1/fgtgroupfilter/[id]/	Update an existing FortiGate Group Filter specified with ID id

Allowed Filters

Field	Filters
shortname	`exact, iexact, contains, icontains, in`

View FortiGate Group Filter Configuration

JSON Query

- JSON specified via GET

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    https://192.168.0.122/api/v1/fgtgroupfilter/?format=json
```

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -H 'Accept:
    application/json' https://192.168.0.122/api/v1/fgtgroupfilter/
```

Response

```
< HTTP/1.1 200 OK
< Date: Tue, 10 Jun 2014 13:49:24 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
{"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 1},
    "objects": [{"address": "1.1.1.1", "id": 1, "name": "GroupFilter_Test1", "nasname":
    "1.1.1.1", "resource_uri": "/api/v1/fgtgroupfilter/1/", "shortname": "GroupFilter_
    Test1", "sso_groups": [
```

Add FortiGate Group Filter Configuration

Note that POST is not an allowed method so FGTGroup filters cannot be created via the API, howver once created via the GUI, they can be modified. See below.

Modify FortiGate Group Filter Configuration

JSON Query

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -X PUT -d '
    {"shortname":"GroupFilter_Test1","nasname":"2.2.2.2", "sso_groups": []}' -H 'Content-
    Type: application/json' https://192.168.0.122/api/v1/fgtgroupfilter/1/
```

Response

```
< HTTP/1.1 204 NO CONTENT
< Date: Mon, 16 Jun 2014 16:35:16 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
```

SSO Auth (/ssoauth/)

URL: `https://[server_name]/api/[api_version]/ssoauth/`

This end-point represents the Fortinet SSO Authentication. In the FortiAuthenticator GUI, this resource corresponds to Fortinet SSO Methods→ SSO. This API is for use by third party authentication systems for dynamic transparent user Single Sign-on to a Fortinet protected network.

Note that before attempting to authenticate, additional configuration is required under Fortinet SSO Methods→ Portal Services → SSO Web Service to select which user directory is to be used for group embellishment.

Supported Fields

Field	Display Name	Type	Required	
event	Event type	integer/string	Yes	0=Login 1=Logout
username	User's username	string	Yes	max length=253
user_ip	User's workstation IP (Calling-Station-Id)	IPv4	Yes	
user_ groups	Groups to send (Fortinet-Group-Name)	string	No	max length=253, list of groups must be separated with "+" character (group name cannot contain a "+" character)

For local users, the user must be part of a local group for successful SSO logon.

External users must have a group passed in via the user_groups field for logon/logoff.

Allowed Methods

HTTP Method	Resource URI	Action
POST	api/v1/ssoauth/	Logon/logoff users to/from FSSO

Response Codes

In addition to the general codes defined in Appendix A –API Response Codes, a POST request to this resource can result in the following return codes:

Code	Response Content	Description
200 OK		FSSO login/logout request has been successfully sent to FSSO (but this doesn't mean that user has been logged-on/off, as the request is done asynchronously and is queued on FSSO side. Factors such as configuration and user not rxisting in LDAP may cause the entry to not populate FSSO).
404 Not Found	SSO web service is disabled	SSO web service has not been enabled so it can't be used in REST API
500 Internal Server Error		Failed to send logon/logoff request to FSSO

FSSO User Login

JSON Query

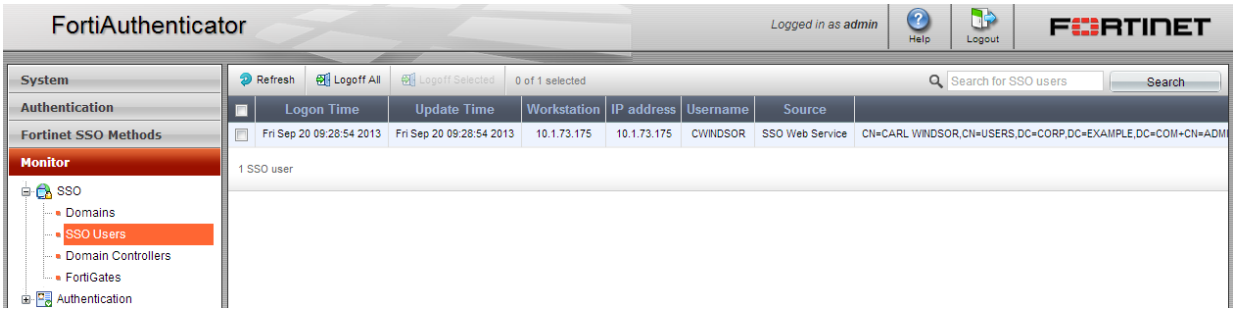
- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -d '
    {"event":"1","username":"cwindsor","user_ip":"10.1.73.175"}' -H "Content-Type:
    application/json" https://192.168.0.122/api/v1/ssoauth/
```

Response

```
< HTTP/1.1 200 OK
< Date: Fri, 20 Sep 2013 08:27:27 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< Content-Language: en
< Set-Cookie: sessionid=6q6m6ne4v7p76qclajitlf2q7202f7g6; httponly; Path=/
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
```

Verify Login on FortiAuthenticator



Overwrite FSSO user login with different user

Note that if a login event is received with the same IP address but with a different username, the existing entry will be overwritten.

JSON Query

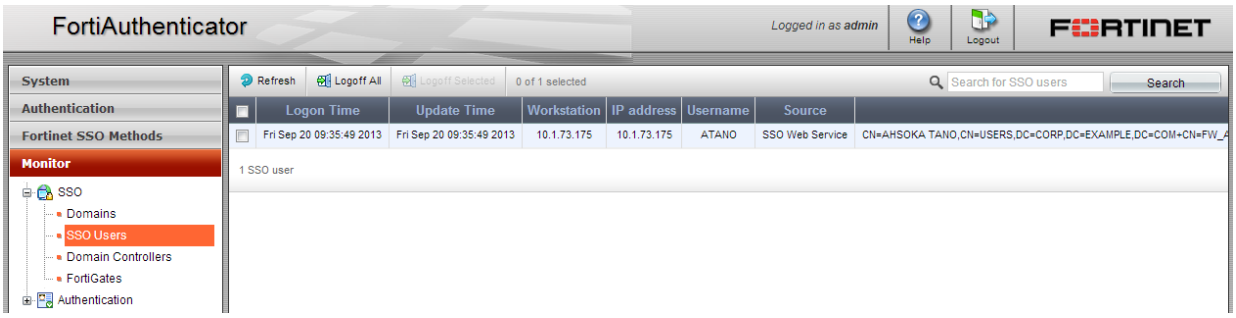
- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -d '
    {"event":"1","username":"atano","user_ip":"10.1.73.175"}' -H "Content-Type:
    application/json" https://192.168.0.122/api/v1/ssoauth/
```

Response

```
< HTTP/1.1 200 OK
< Date: Fri, 20 Sep 2013 08:32:21 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< Content-Language: en
< Set-Cookie: sessionid=g062qqmsj6nr0hk5khd2q7202e4v36m; httponly; Path=/
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
```

Verify Login on FortiAuthenticator



Logout FSSO User

JSON Query

- JSON specified via Accept Header

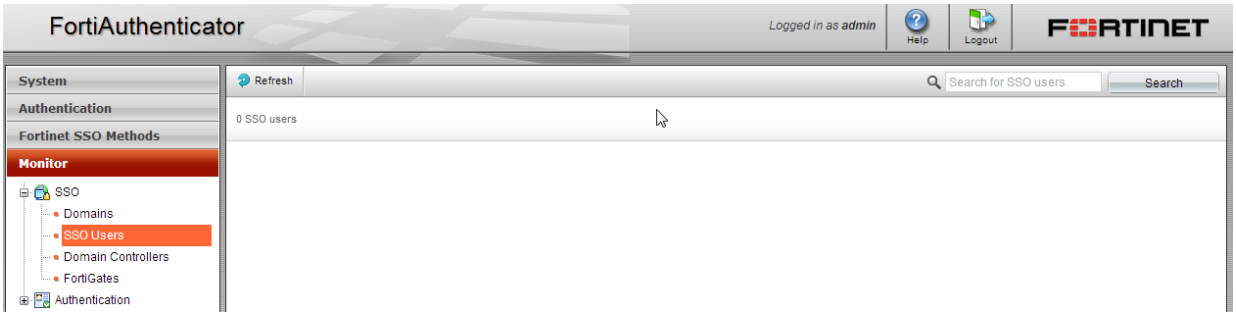
```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -d '
    {"event":"0","username":"atano","user_ip":"10.1.73.175"}' -H "Content-Type:
    application/json" https://192.168.0.122/api/v1/ssoauth/
```

Response

```
< HTTP/1.1 200 OK
< Date: Fri, 20 Sep 2013 08:34:09 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< Content-Language: en
```

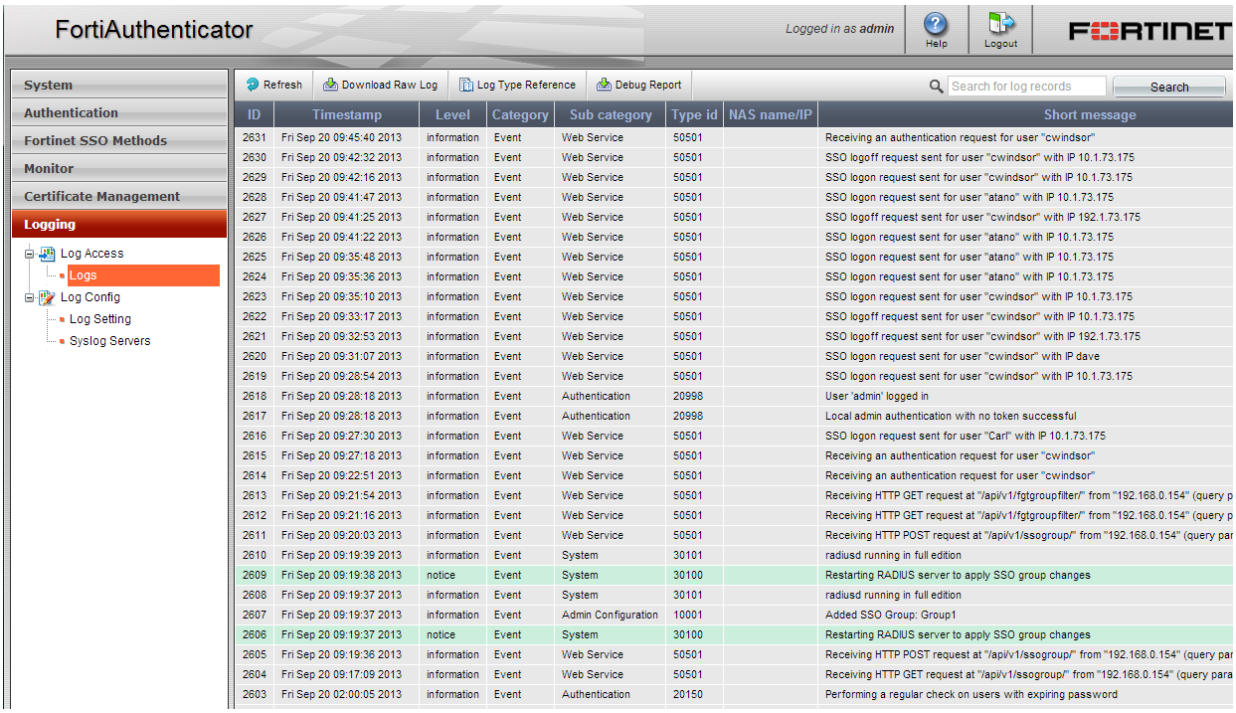
```
< Set-Cookie: sessionid=2q de4v36msj6g05khm6nr02q72q02hk; httponly; Path=/
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
```

Verify Logout on FortiAuthenticator



Logging

Note that SSO Login requests are logged regardless of whether the user details can be inserted into FSSO. For example logs may exist for SSO Logon for a user but an entry not appear in the monitor because when an LDAP lookup for group info was performed, no user existed.



SSO Filtering Objects (/fgtgroupfilter/[id]/ssofilterobjects/)

URL: `https://[server_name]/api/v1/fgtgroupfilter/[id]/ssofilterobjects/`

This resource can only be used along side FortiGate Filter resource above.

Supported Fields

Field	Display Name	Type	Required	Other Restriction
name	Object name / DN	string	Yes	max length=255, unique for each FortiGate filter
obj_type	Object Type	string	Yes	One of user, group (default), user container, group container, user and group container

Allowed Methods

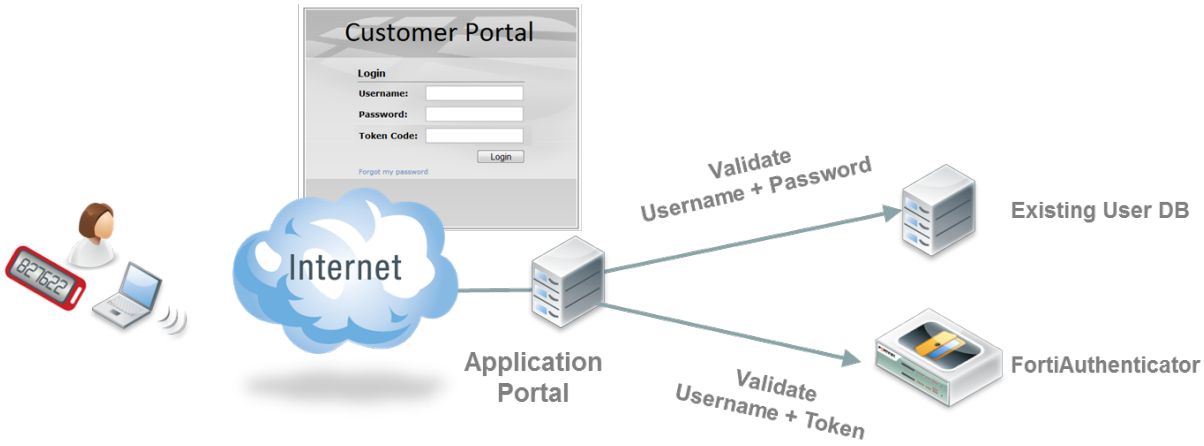
HTTP Method	Resource URI	Action
GET	/api/v1/fgtgroupfilter/[id] /ssofilterobjects/	Get all SSO filtering objects for a specific FortiGate filter
GET	/api/v1/fgtgroupfilter/[id] /ssofilterobjects/[filter_id]/	Get an SSO filtering object for a specific FortiGate filter
POST	/api/v1/fgtgroupfilter/[id] /ssofilterobjects/	Create a new SSO filtering object for a specific FortiGate filter
PUT	/api/v1/fgtgroupfilter/[id] /ssofilterobjects/	Update all SSO filtering objects that belongs to a FortiGate filter
PATCH	/api/v1/fgtgroupfilter/[id] /ssofilterobjects/[filter_id]/	Update fields of an SSO filtering object
DELETE	/api/v1/fgtgroupfilter/[id] /ssofilterobjects/	Delete all SSO filtering objects from a specific FortiGate filter
DELETE	/api/v1/fgtgroupfilter/[id] /ssofilterobjects/[filter_id]/	Delete an SSO filtering object

Authentication (/auth/)

URL: `https://[server_name]/api/[api_version]/auth/`

The Authentication API is for validation of user credentials. Either the password, token or both can be validated. This is useful for adding an additional factor authentication (e.g. token) to web portals where the first factor as already being validated locally e.g. via LDAP or local DB or a proprietary, unsupported authentication method as is common in the banking industry.

This API is for the validation of local user password and token passcode or remote user passcode only. Validation of remote (LDAP) user password is not supported. This is by design as most systems have an established mechanism for authentication via e.g. LDAP or some other proprietary mechanism as shown below.



To authenticate a user, you need to POST to https://[server_name]/api/1/auth/ with the following key-value pair (in JSON format, but XML also possible):

```
{"username": "<username>", "token_code": "<token_code>", "password":
    "<password>"}
```

with "token_code" and "password" being optional fields i.e. you can just validate the token only or the password only. If password and token are specified, the password will be validated first before token code.

Behavior of the API

Either password or token_code needs to be specified.

- If both are specified, password will be validated first, then token_code.
- If only one is specified (either password or token_code), only that credential will be validated.
- If a user doesn't have two-factor authentication configured, validation for that user with any token_code will fail.

Supported Fields

Field	Display Name	Type	Required	Other Restriction
username	Username	string	Yes	
password	Password	string	No	
token_code	Security token code	string	No	Supported token authentication: FortiToken, email token, SMS token

Allowed Methods

Type	Allowed Methods	Action
List	POST	Validate user's credentials

Response Codes

In addition to the general codes defined in Appendix A –API Response Codes, a POST request to this resource can result in the following return codes:

Code	Response Content	Description
200 OK		User is successfully authenticated.
401 Unauthorized	User authentication failed	Credential is incorrect
401 Unauthorized	Account is disabled	User account is currently disabled
401 Unauthorized	No token configured	User does not have token-based authentication configured
401 Unauthorized	Token is out of sync	The security token requires synchronization
404 Not Found	User does not exist	The given username does not exist in the system

Validate a user password

Query

- JSON specified via Accept Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -d '
    {"username":"testuser","password":"testpass"}' -H "Content-Type: application/json"
    https://192.168.0.122/api/v1/auth/
```

Response

```
< HTTP/1.1 200 OK
< Date: Fri, 14 Sep 2012 15:38:57 GMT
< Server: Apache
< Vary: Cookie
< Set-Cookie: sessionid=6b17c5bbb86419a94f6979a05bd84139; httponly; Path=/
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
```

Validate a users token code

Query

- JSON specified via Content-Type Header

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS" -d '
    {"username":"testuser","token_code":"893753"}' -H "Content-Type: application/json"
    https://192.168.0.122/api/v1/auth/
```

Response

```
< HTTP/1.1 200 OK
< Date: Fri, 14 Sep 2012 15:47:22 GMT
< Server: Apache
< Vary: Cookie
< Set-Cookie: sessionid=f15beeab159a4bf2d0402a05db40d6ae; httponly; Path=/
< Content-Length: 0
< Content-Type: text/html; charset=utf-8
```

Error States

Response (incorrect password)

```
HTTP/1.1 401 UNAUTHORIZED
Date: Thu, 13 Sep 2012 13:57:24 GMT
Server: Apache
Vary: Cookie
Set-Cookie: sessionid=abe8bac6fc50caf5eadf1e57f0c60e3e; httponly; Path=/
Content-Length: 26
Content-Type: text/html; charset=utf-8
```

Response (incorrect token code)

```
HTTP/1.1 401 UNAUTHORIZED
Date: Thu, 13 Sep 2012 13:55:18 GMT
Server: Apache
Vary: Cookie
Set-Cookie: sessionid=e95090804ee0e3b8903618138b38a5c8; httponly; Path=/
Content-Length: 26
Content-Type: text/html; charset=utf-8
```

Response (incorrect username)

```
HTTP/1.1 404 NOT FOUND
Date: Thu, 13 Sep 2012 13:58:54 GMT
Server: Apache
Vary: Cookie
```

```
Set-Cookie: sessionid=3b353061d9141567c02bb0d057b18284; httponly; Path=/
Content-Length: 19
Content-Type: text/html; charset=utf-8
```

Advanced Filtering

Results of the API calls can be controlled in several ways. Below are some arguments that can be passed to the REST API URL. Please refer to the specific resource documentation to find out which of these filter operations are allowed.

General Filters

General filters can be applied to most resources.

Limits

limit: *limit number of items returned*

To search for the first entry in a resource

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    “https://192.168.0.122/api/v1/localusers/?format=json&limit=1”
```

The URL requires additional quoting in this case otherwise the Unix CLI treats the “&” as a instruction to place the cURL command into the background.

Response

```
< HTTP/1.1 200 OK
< Date: Tue, 10 Jun 2014 09:43:33 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
* Connection #0 to host 192.168.0.122 left intact
* Closing connection #0
{"meta": {"limit": 1, "next": "/api/v1/localusers/?offset=1&limit=1&format=json",
    "offset": 0, "previous": null, "total_count": 3}, "objects": [{"address": "", "city":
    "", "country": "", "custom1": "", "custom2": "", "custom3": "", "email": "", "first_
    name": "", "id": 5, "last_name": "", "mobile_number": "", "phone_number": "",
    "resource_uri": "/api/v1/localusers/5/", "state": "", "token_auth": false, "token_
    serial": "", "token_type": null, "user_groups": ["/api/v1/usergroups/9/",
    "/api/v1/usergroups/8/"], "username": "test_user2"}]}
```

Only the first user in the list is returned. Note that this excludes admin users which are never returned by this query hence the reason why this user ID is > 5.

Offset

offset: specify an offset for the returned items (zero-based). E.g. if there are 10 items, to return item #5 - #10 only, specify offset=4

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    "https://192.168.0.122/api/v1/localusers/?format=json&offset=4"
```

Order

order_by: *order returned list by a known field name (e.g. ?order_by=<field name>)*

```
curl -k -v -u "admin:zeyDZXmP6GbKcerqdWWEYNTnH2TaOCz5HTp2dAVS"
    "https://192.168.0.122/api/v1/localusers/?format=json&order_by=username"
```

Response

```
< HTTP/1.1 200 OK
< Date: Tue, 10 Jun 2014 16:41:23 GMT
< Server: Apache
< Vary: Accept,Accept-Language,Cookie
< X-Frame-Options: SAMEORIGIN
< Content-Language: en
< Cache-Control: no-cache
< Transfer-Encoding: chunked
< Content-Type: application/json
<
{"meta": {"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 3},
    "objects": [{"address": "", "city": "", "country": "", "custom1": "", "custom2": "",
    "custom3": "", "email": "", "first_name": "", "id": 4, "last_name": "", "mobile_
    number": "", "phone_number": "", "resource_uri": "/api/v1/localusers/4/", "state": "",
    "token_auth": false, "token_serial": "", "token_type": null, "user_groups":
    ["/api/v1/usergroups/8/"], "username": "test_user"}, {"address": "", "city": "",
    "country": "GB", "custom1": "example", "custom2": "", "custom3": "", "email": "",
    "first_name": "", "id": 5, "last_name": "", "mobile_number": "", "phone_number": "",
    "resource_uri": "/api/v1/localusers/5/", "state": "", "token_auth": false, "token_
    serial": "", "token_type": null, "user_groups": ["/api/v1/usergroups/9/",
    "/api/v1/usergroups/8/"], "username": "test_user2"}, {"address": "", "city": "",
    "country": "GB", "custom1": "example", "custom2": "", "custom3": "", "email": "test_
    user3@example.com", "first_name": "", "id": 6, "last_name": "", "mobile_number": "",
    "phone_number": "", "resource_uri": "/api/v1/localusers/6/", "state": "", "token_
    auth": false, "token_serial": "", "token_type": null, "user_groups": [], "username":
    "test_user3"}]}
```

Filter Lookup Expressions

Expression	Description
`exact`	search for an exact match (e.g. `name__exact=John Doe`, would return user with name "John Doe", but not "john doe")
`iexact`	search for a case-insensitive exact match (e.g. `name__iexact=john doe`, would return user with name "John Doe")

Expression	Description
`contains`	search for an item that contains a specific keyword
`icontains`	same as above, but case-insensitive
`in`	search for items that matches specific filter criteria (e.g. to return items that has a name matching "John" or "Bill", `?name__in=John&name__in=Bill`)
`startswith`	search for items that starts with a text
`istartswith`	same as above, but case-insensitive

Appendix A –API Response Codes

General API Response Codes

Code	Description
200 OK	The request was successfully completed.
201 Created	The request successfully created a new resource and the response body does not contain the newly created resource.
202 Accepted	The server fulfilled the request and the response body contains the newly updated resource.
204 No Content	The server fulfilled the request, but does not need to return a response message body.
400 Bad Request	The request could not be processed because it contains missing or invalid information (i.e. the data in the request does not validate).
401 Not Authorized	The supplied credential is incorrect.
403 Forbidden	Permission is denied to perform an operation.
500 Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.